US 20020129212A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0129212 A1

Lee et al. (43) Pub. Date: Sep. 12, 2002

(54) VIRTUALIZED NVRAM ACCESS METHODS TO PROVIDE NVRAM CHRP REGIONS FOR LOGICAL PARTITIONS THROUGH HYPERVISOR SYSTEM CALLS

(75) Inventors: **Van Hoa Lee**, Cedar Park, TX (US); **Kanisha Patel**, Cedar Park, TX (US); **David R. Willoughby**, Austin, TX (US)

Correspondence Address:
Duke W. Yee
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380 (US)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY

(21) Appl. No.: **09/798,292**

(22) Filed: **Mar. 1, 2001**

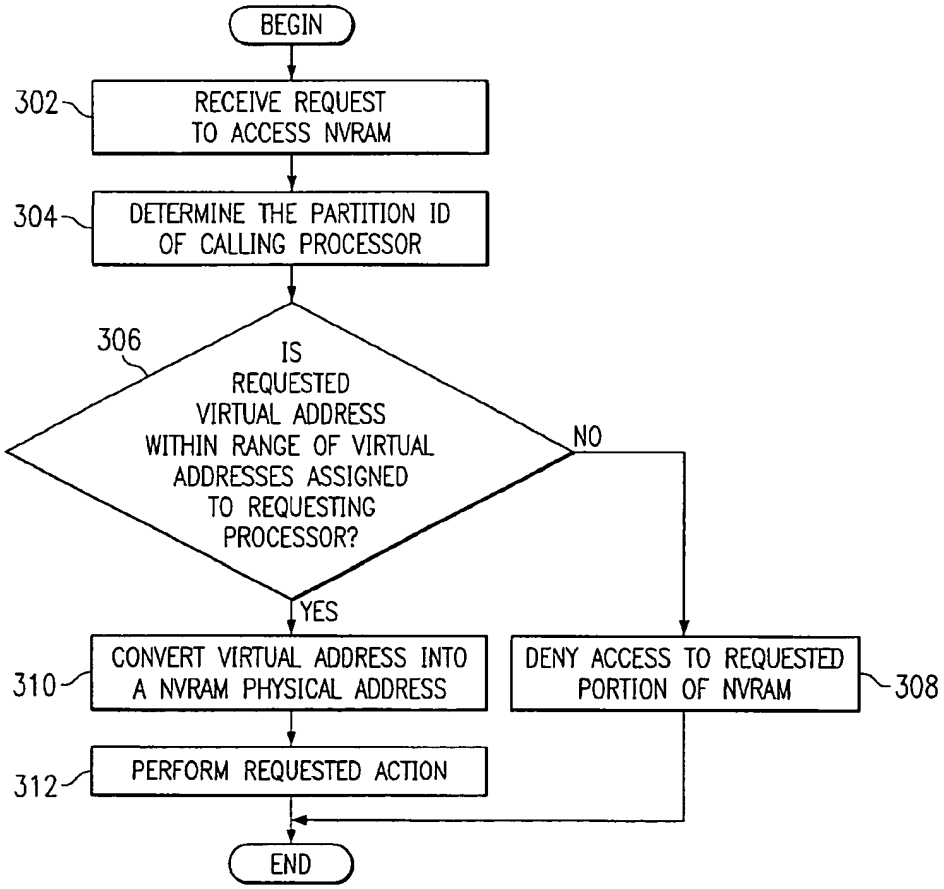
Publication Classification

(51) **Int. Cl.**[7] **G06F 12/14**

(52) **U.S. Cl.** **711/152**; 711/173

(57) **ABSTRACT**

A method, system, and computer program product for enforcing logical partitioning of a shared device to which multiple partitions within a data processing system have access is provided. In one embodiment, a firmware portion of the data processing system receives a request from a requesting device, such as a processor assigned to one of a plurality of partitions within the data processing system, to access (i.e., read from or write to) a portion of the shared device, such as an NVRAM. The request includes a virtual address corresponding to the portion of the shared device for which access is desired. If the virtual address is within a range of addresses for which the requesting device is authorized to access, the firmware provides access to the requested portion of the shared device to the requesting device. If the virtual address is not within a range of addresses for which the requesting device is authorized to access, the firmware denies the request.



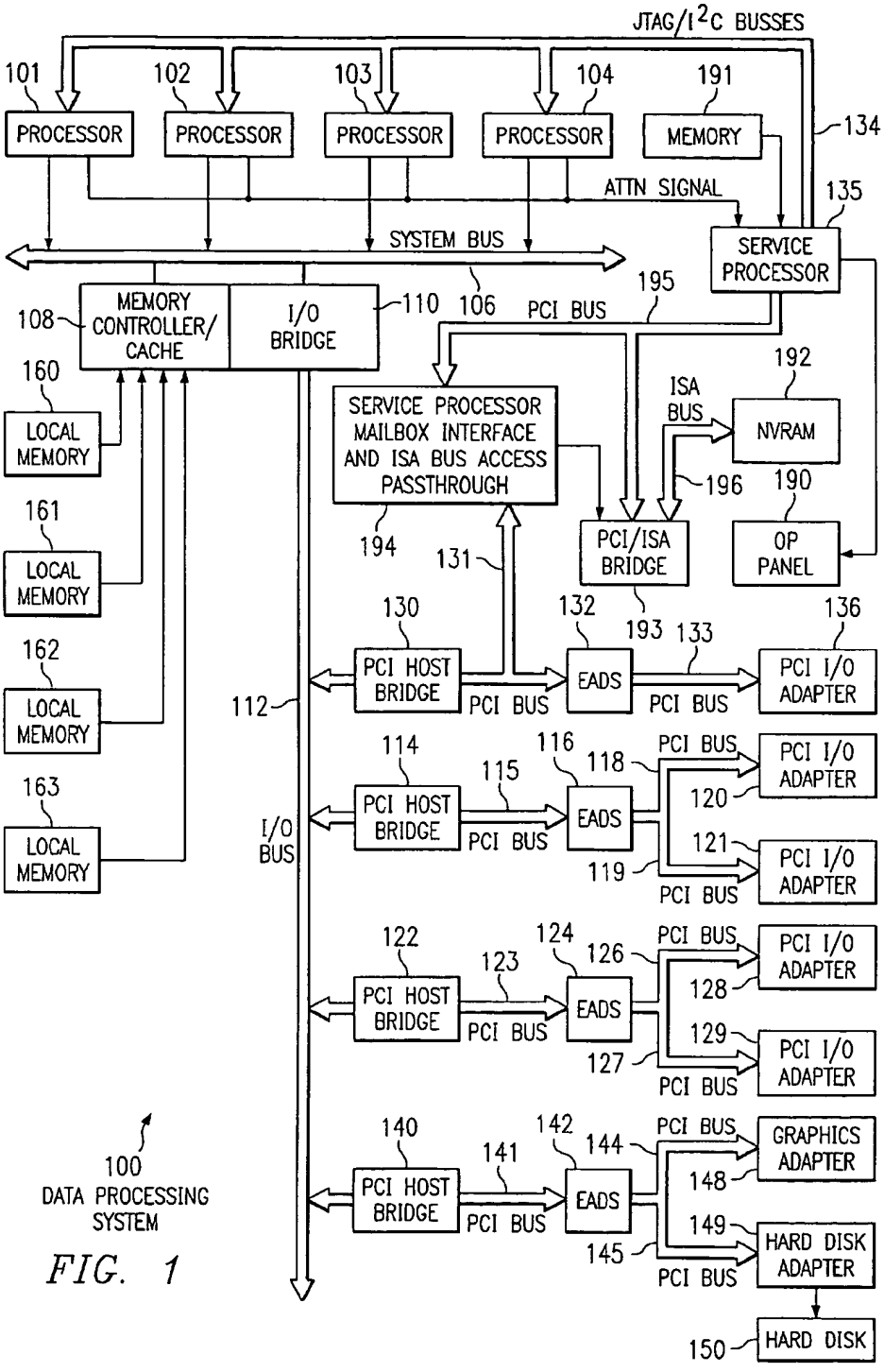

JTAG/I2C BUSSES
101
102
103
104
191
PROCESSOR
PROCESSOR
PROCESSOR
PROCESSOR
MEMORY
134
ATTN SIGNAL
135
SYSTEM BUS
SERVICE PROCESSOR
108
MEMORY CONTROLLER/ CACHE
I/O BRIDGE
110
106
PCI BUS
195
192
160
ISA BUS
SERVICE PROCESSOR MAILBOX INTERFACE AND ISA BUS ACCESS PASSTHROUGH
NVRAM
LOCAL MEMORY
196
190
161
194
131
PCI/ISA BRIDGE
OP PANEL
LOCAL MEMORY
130
132
193
133
136
162
PCI HOST BRIDGE
EADS
PCI I/O ADAPTER
LOCAL MEMORY
112
PCI BUS
PCI BUS
114
116
118
PCI BUS
163
115
PCI HOST BRIDGE
EADS
PCI I/O ADAPTER
LOCAL MEMORY
I/O BUS
PCI BUS
120
121
PCI I/O ADAPTER
119
PCI BUS
122
124
126
PCI BUS
PCI HOST BRIDGE
123
EADS
PCI I/O ADAPTER
128
PCI BUS
129
PCI I/O ADAPTER
127
PCI BUS
140
142
144
PCI BUS
GRAPHICS ADAPTER
PCI HOST BRIDGE
141
EADS
148
PCI BUS
149
HARD DISK ADAPTER
145
PCI BUS
150
HARD DISK
100
DATA PROCESSING SYSTEM

FIG. 1

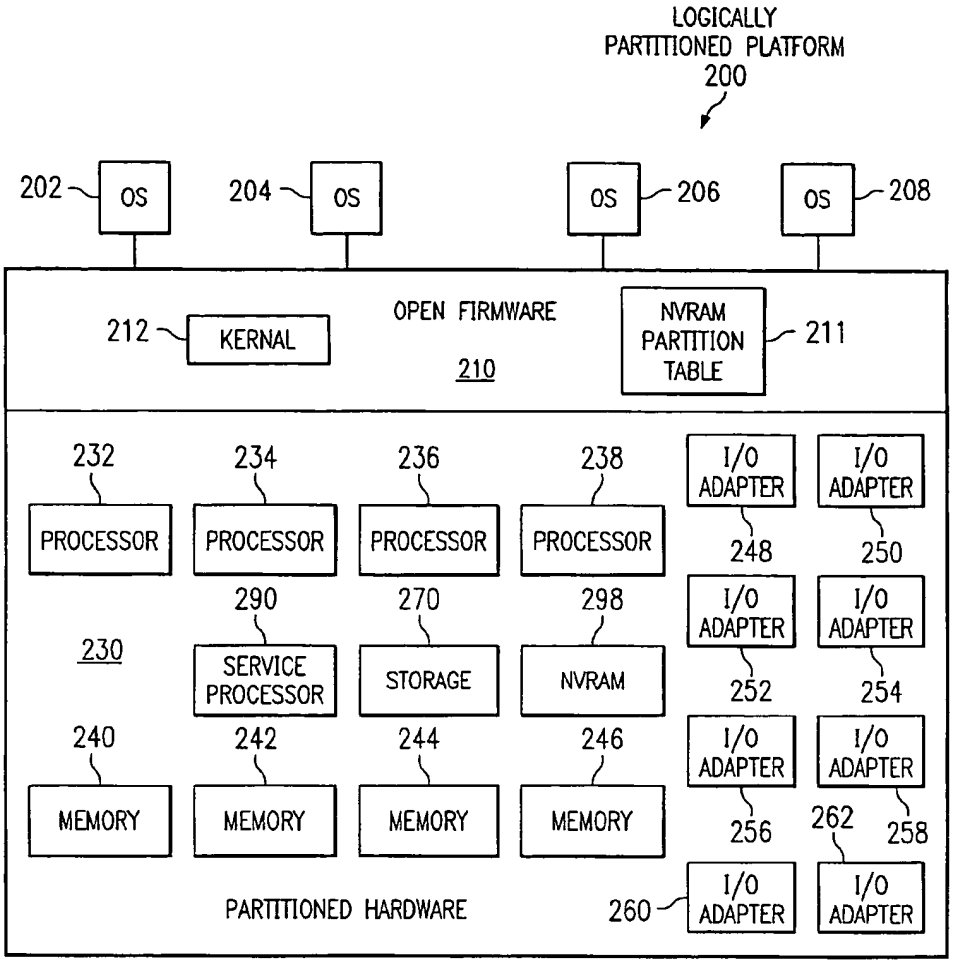
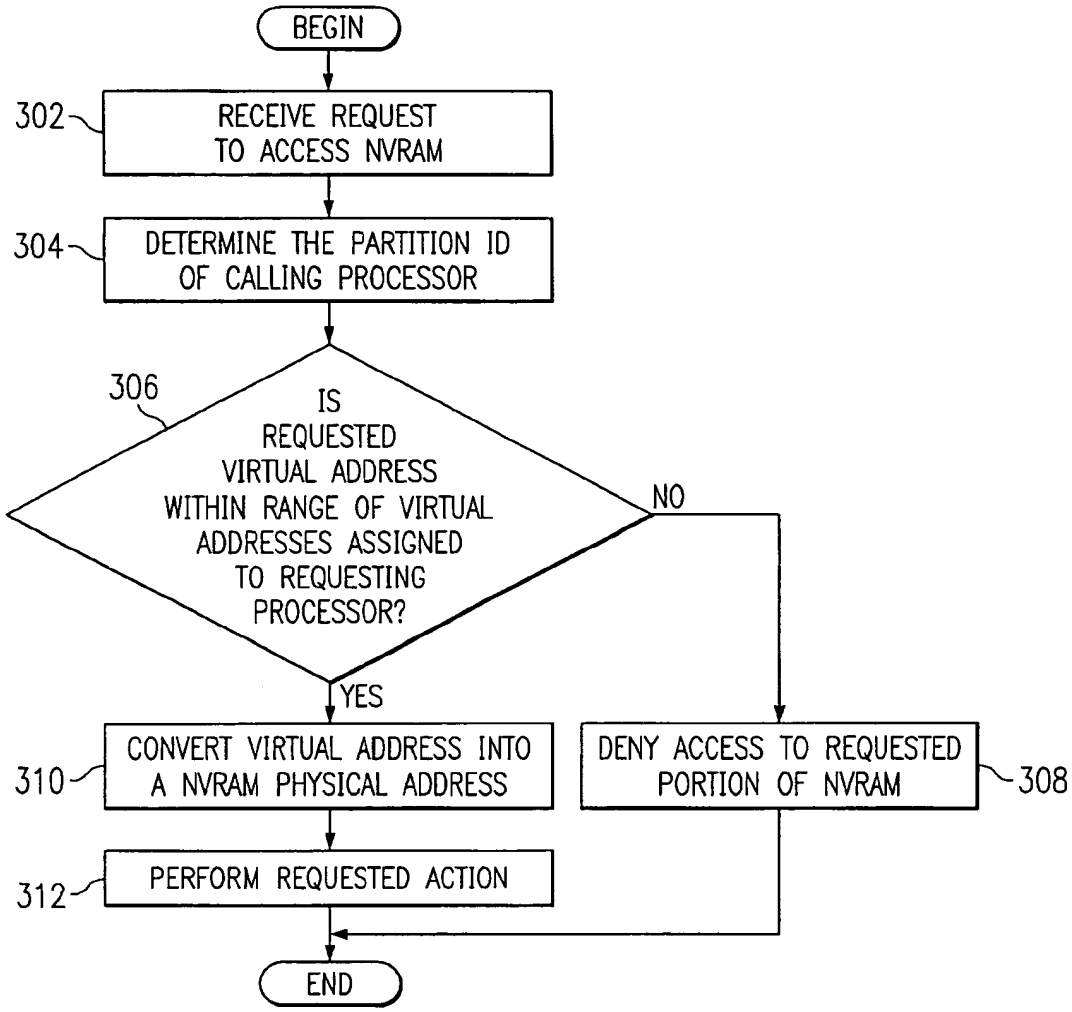

LOGICALLY
PARTITIONED PLATFORM
200
202
OS
204
OS
OS
206
OS
208
212
KERNAL
OPEN FIRMWARE
210
NVRAM
PARTITION
TABLE
211
232
234
236
238
PROCESSOR
PROCESSOR
PROCESSOR
PROCESSOR
I/O
ADAPTER
I/O
ADAPTER
248
250
290
270
298
230
SERVICE
PROCESSOR
STORAGE
NVRAM
I/O
ADAPTER
I/O
ADAPTER
252
254
240
242
244
246
MEMORY
MEMORY
MEMORY
MEMORY
I/O
ADAPTER
I/O
ADAPTER
256
262
258
PARTITIONED HARDWARE
260
I/O
ADAPTER
I/O
ADAPTER

FIG. 2

FIG. 3


BEGIN
302
RECEIVE REQUEST TO ACCESS NVRAM
304
DETERMINE THE PARTITION ID OF CALLING PROCESSOR
306
IS REQUESTED VIRTUAL ADDRESS WITHIN RANGE OF VIRTUAL ADDRESSES ASSIGNED TO REQUESTING PROCESSOR?
NO
YES
310
CONVERT VIRTUAL ADDRESS INTO A NVRAM PHYSICAL ADDRESS
DENY ACCESS TO REQUESTED PORTION OF NVRAM
308
312
PERFORM REQUESTED ACTION
END

VIRTUALIZED NVRAM ACCESS METHODS TO PROVIDE NVRAM CHRP REGIONS FOR LOGICAL PARTITIONS THROUGH HYPERVISOR SYSTEM CALLS

BACKGROUND OF THE INVENTION

[0001] 1. Technical Field:

[0002] The present invention relates generally to an improved data processing system and, more particularly, to an improved logically partitioned data processing system. Still more particularly, the present invention relates to an improved non-volatile random access memory access for logically partitioned data processing systems.

[0003] 2. Description of Related Art:

[0004] A logical partitioning (LPAR) functionality within a data processing system (platform) allows multiple copies of a single operating system (OS) or multiple heterogeneous operating systems to be simultaneously run on a single data processing system platform. A partition, within which an operating system image runs, is assigned a non-overlapping sub-set of the platform's resources. These platform allocable resources include one or more architecturally distinct processors with their interrupt management area, regions of system memory, and I/O adapter bus slots. The partition's resources are represented by the platform's firmware to the OS image.

[0005] Each distinct OS or image of an OS running within the platform are protected from each other such that software errors on one logical partition cannot affect the correct operation of any of the other partitions. This is provided by allocating a disjoint set of platform resources to be directly managed by each OS image and by providing mechanisms for ensuring that the various images cannot control any resources that have not been allocated to it. Furthermore, software errors in the control of an OS's allocated resources are prevented from affecting the resources of any other image. Thus, each image of the OS (or each different OS) directly controls a distinct set of allocable resources within the platform.

[0006] Currently, some resources existing singly within the data processing system are shared by more than one partition. It would be desirable to have a mechanism by which these single resources may be logically partitioned and have the logical partitioning strictly enforced.

SUMMARY OF THE INVENTION

[0007] The present invention provides a method, system, and computer program product for enforcing logical partitioning of a shared device to which multiple partitions within a data processing system have access. In one embodiment, a firmware portion of the data processing system receives a request from a requesting device, such as a processor assigned to one of a plurality of partitions within the data processing system, to access (i.e., read from or write to) a portion of the shared device, such as an NVRAM. The request includes a virtual address corresponding to the portion of the shared device for which access is desired. If the virtual address is within a range of addresses for which the requesting device is authorized to access, the firmware provides access to the requested portion of the shared device to the requesting device. If the virtual address is not within a range of addresses for which the requesting device is authorized to access, the firmware denies the request.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

[0009] **FIG. 1** depicts a block diagram of a data processing system in which the present invention may be implemented;

[0010] **FIG. 2** depicts a block diagram of an exemplary logically partitioned platform in which the present invention may be implemented; and

[0011] **FIG. 3** depicts a flowchart illustrating an exemplary method for enforcing logical partitioning within a non-volatile random access memory in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0012] With reference now to the figures, and in particular with reference to **FIG. 1**, a block diagram of a data processing system in which the present invention may be implemented is depicted. Data processing system **100** may be a symmetric multiprocessor (SMP) system including a plurality of processors **101**, **102**, **103**, and **104** connected to system bus **106**. For example, data processing system **100** may be an IBM RS/6000, a product of International Business Machines Corporation in Armonk, N.Y., implemented as a server within a network. Alternatively, a single processor system may be employed. Also connected to system bus **106** is memory controller/cache **108**, which provides an interface to a plurality of local memories **160-163**. I/O bus bridge **110** is connected to system bus **106** and provides an interface to I/O bus **112**. Memory controller/cache **108** and I/O bus bridge **110** may be integrated as depicted.

[0013] Data processing system **100** is a logically partitioned data processing system. Thus, data processing system **100** may have multiple heterogeneous operating systems (or multiple instances of a single operating system) running simultaneously. Each of these multiple operating systems may have any number of software programs executing within in it. Data processing system **100** is logically partitioned such that different I/O adapters **120-121**, **128-129**, **136**, and **148-149** may be assigned to different logical partitions. Furthermore, NVRAM **192** may be partitioned such that each partition may access and utilize only certain portions of NVRAM **192**.

[0014] Thus, for example, suppose data processing system **100** is divided into three logical partitions, P1, P2, and P3. Each of I/O adapters **120-121**, **128-129**, **136**, and **148-149**, each of processors **101-104**, and each of local memories **160-164** is assigned to one of the three partitions. For example, processor **101**, memory **160**, and I/O adapters **120**, **128**, and **129** may be assigned to logical partition P1; processors **102-103**, memory **161**, and I/O adapters **121** and **136** may be assigned to partition P2; and processor **104**,

memories **162-163**, and I/O adapters **148-149** may be assigned to logical partition P3.

[0015] Each operating system executing within data processing system **100** is assigned to a different logical partition. Thus, each operating system executing within data processing system **100** may access only those I/O units that are within its logical partition. Thus, for example, one instance of the Advanced Interactive Executive (AIX) operating system may be executing within partition P1, a second instance (image) of the AIX operating system may be executing within partition P2, and a Windows 2000 operating system may be operating within logical partition P1. Windows 2000 is a product and trademark of Microsoft Corporation of Redmond, Wash.

[0016] Peripheral component interconnect (PCI) Host bridge **114** connected to I/O bus **112** provides an interface to PCI local bus **115**. A number of Input/Output adapters **120-121** may be connected to PCI bus **115**. Typical PCI bus implementations will support between four and eight I/O adapters (i.e. expansion slots for add-in connectors). Each I/O Adapter **120-121** provides an interface between data processing system **100** and input/output devices such as, for example, other network computers, which are clients to data processing system **100**.

[0017] An additional PCI host bridge **122** provide an interface for an additional PCI bus **123**. PCI bus **123** is connected to a plurality of PCI I/O adapters **128-129** by a PCI bus **126-127**. Thus, additional I/O devices, such as, for example, modems or network adapters may be supported through each of PCI I/O adapters **128-129**. In this manner, data processing system **100** allows connections to multiple network computers.

[0018] A memory mapped graphics adapter **148** may be connected to I/O bus **112** through PCI Host Bridge **140** and EADS **142** (PCI-PCI bridge) via PCI buses **141** and **144** as depicted. Also, a hard disk **150** may also be connected to I/O bus **112** through PCI Host Bridge **140** and EADS **142** via PCI buses **141** and **145** as depicted.

[0019] A PCI host bridge **130** provides an interface for a PCI bus **131** to connect to I/O bus **112**. PCI bus **131** connects PCI host bridge **130** to the service processor mailbox interface and ISA bus access pass-through logic **194** and EADS **132**. The ISA bus access pass-through logic **194** forwards PCI accesses destined to the PCI/ISA bridge **193**. The NV-RAM storage is connected to the ISA bus **196**. The Service processor **135** is coupled to the service processor mailbox interface **194** through its local PCI bus **195**. Service processor **135** is also connected to processors **101-104** via a plurality of JTAG/I^2C buses **134**. JTAG/I^2C buses **134** are a combination of JTAG/scan busses (see IEEE 1149.1) and Phillips I^2C busses. However, alternatively, JTAG/I^2C buses **134** may be replaced by only Phillips I^2C busses or only JTAG/scan busses. All SP-ATTN signals of the host processors **101**, **102**, **103**, and **104** are connected together to an interrupt input signal of the service processor. The service processor **135** has its own local memory **191**, and has access to the hardware op-panel **190**.

[0020] When data processing system **100** is initially powered up, service processor **135** uses the JTAG/scan buses **134** to interrogate the system (Host) processors **101-104**, memory controller **108**, and I/O bridge **110**. At completion of this step, service processor **135** has an inventory and topology understanding of data processing system **100**. Service processor **135** also executes Built-In-Self-Tests (BISTs), Basic Assurance Tests (BATs), and memory tests on all elements found by interrogating the system processors **101-104**, memory controller **108**, and I/O bridge **110**. Any error information for failures detected during the BISTs, BATs, and memory tests are gathered and reported by service processor **135**.

[0021] If a meaningful/valid configuration of system resources is still possible after taking out the elements found to be faulty during the BISTS, BATs, and memory tests, then data processing system **100** is allowed to proceed to load executable code into local (Host) memories **160-163**. Service processor **135** then releases the Host processors **101-104** for execution of the code loaded into Host memory **160-163**. While the Host processors **101-104** are executing code from respective operating systems within the data processing system **100**, service processor **135** enters a mode of monitoring and reporting errors. The type of items monitored by service processor include, for example, the cooling fan speed and operation, thermal sensors, power supply regulators, and recoverable and non-recoverable errors reported by processors **101-104**, memories **160-163**, and bus-bridge controller **110**.

[0022] Service processor **135** is responsible for saving and reporting error information related to all the monitored items in data processing system **100**. Service processor **135** also takes action based on the type of errors and defined thresholds. For example, service processor **135** may take note of excessive recoverable errors on a processor's cache memory and decide that this is predictive of a hard failure. Based on this determination, service processor **135** may mark that resource for deconfiguration during the current running session and future Initial Program Loads (IPLs). IPLs are also sometimes referred to as a "boot" or "bootstrap".

[0023] Those of ordinary skill in the art will appreciate that the hardware depicted in **FIG. 1** may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

[0024] With reference now to **FIG. 2**, a block diagram of an exemplary logically partitioned platform is depicted in which the present invention may be implemented. The hardware in logically partitioned platform **200** may be implemented as, for example, server **100** in **FIG. 1**. Logically partitioned platform **200** includes partitioned hardware **230**, Open Firmware (OF) **210**, and operating systems **202-208**. OF **210** is sometimes referred to as a hypervisor in some International Business Machine implementations. Operating systems **202-208** may be multiple copies of a single operating system or multiple heterogeneous operating systems simultaneously run on platform **200**.

[0025] Partitioned hardware **230** includes a plurality of processors **232-238**, a plurality of system memory units **240-246**, a plurality of input/output (I/O) adapters **248-262**, and a storage unit **270**. Each of the processors **242-248**, memory units **240-246**, and I/O adapters **248-262** may be assigned to one of multiple partitions within logically partitioned platform **200**, each of which corresponds to one of operating systems **202-208**.

[0026] OF **210** performs a number of functions and services for operating system images **202-208** to create and enforce the partitioning of logically partitioned platform **200**. Firmware is "software" stored in a memory chip that holds its content without electrical power, such as, for example, read-only memory (ROM), programmable ROM (PROM), erasable programmable ROM (EPROM), electrically erasable programmable ROM (EEPROM), and non-volatile random access memory (non-volatile RAM).

[0027] OF **210** is a firmware implemented virtual machine identical to the underlying hardware. Thus, OF **210** allows the simultaneous execution of independent OS images **202-208** by virtualizing all the hardware resources of logically partitioned platform **200**. OF **210** may attach I/O devices through I/O adapters **248-262** to single virtual machines in an exclusive mode for use by one of OS images **202-208**.

[0028] Under logical partitioning, the common NVRAM **298** system resource is divided into several small Common Hardware Reference Platform (CHRP) regions, one each for each of the partitions. CHRP is a platform system architecture, the goal of which is to support multiple Operating Systems on the same hardware platform. The CHRP architecture requires the system platform to provide a portiton of NVRAM storage for the OS running on the system. The OS's NVRAM storage is said to be the CHRP NVRAM region. Since there is no hardware protection to detect unwanted accesses from one partition to another partition's NVRAM CHRP region, all accesses are performed through OF **210** system calls. OF **210** therefore provides a virtualized NVRAM CHRP region for all partitions. However, a partition with service authority may gain access to other NVRAM regions in addition to its own NVRAM CHRP region.

[0029] A normal partition has a virtualized NVRAM address range from zero to (chrp_size−1) with the CHRP region of chrp size bytes. The service authorized partition will have a virtualized NVRAM address range from zero to (chrp_size+auth_size−1) where chrp_size is CHRP region size in bytes and auth_size is Service Authority region size in bytes.

[0030] Any partition can issue NVRAM read/write accesses in its virtual address range. Accessing violating protection is enforced by OF **210** and any partition request for access to a virtual range outside the range allocated to the partition will be rejected by OF **210**. For good accesses, the virtual NVRAM addresses are converted into real NVRAM addresses to read or write from or to the NVRAM **298**.

[0031] OF **210** maintains a chip_base_address of each partition's CHRP region in NVRAM partition table **211**. OF **210** uses the partition ID of the calling processor **234-238** to select the proper chrp_base_address of the partition's CHRP region. Therefore, for a normal partition, the actual physical NVRAM address (real_nvram_address) is given by the following equation:

real_nvram_address=partition_chrp_base_address+
virtual_nvram_address

[0032] With the service_authorized partition, the NVRAM CHRP region and the service authority region may not be physically contiguous regions within NVRAM **298**. If the virtual_nvram_address is greater than zero but less than the service_autho_partition_size, then the real_nvram_address is equal to the service_autho_base_address plus the virtual_nvram_address. Otherwise, if the service_autho_partition_size is less than or equal to the virtual_nvram_address and if the virtual_nvram_address is less than the chrp_size plus the service_autho_partition_size then the real_nvram_address is equal to (virtual_nvram_address−service_autho_partition_size+partition_chrp_base_address). Preferably, the service authority region is mapped before the partition's CHRP region. This allows the CHRP region to grow as needed.

[0033] With reference now to **FIG. 3**, a flowchart illustrating an exemplary method for enforcing logical partitioning within a non-volatile random access memory is depicted in accordance with the present invention.

[0034] The present method may be implemented by, for example, OF **210** in **FIG. 2**. To begin, the OF receives a request to access the NVRAM from a processor within the data processing system (step **302**). This request may be a request to read from a section of the NVRAM or a request to write to a section of the NVRAM and includes the virtual address or addresses of the portions of the NVRAM for which the processor desires access.

[0035] The OF then determines the partition Identification (ID) of the calling processor (step **304**) and whether the requested virtual address or addresses are within the range of virtual addresses assigned to the partition to which the requesting processor belongs (step **306**). If the virtual addresses are not within the range of virtual addresses assigned to the partition to which the processor belongs, then access to the requested portions of the NVRAM is denied (step **308**). If, however, the requested virtual address or addresses are within the range of virtual addresses assigned to the partition to which the requesting processor belongs, then the virtual address or addresses is converted into a NVRAM physical address (step **310**). The OF then performs the requested action (step **312**) of either reading from or writing to the requested portions of the NVRAM.

[0036] The method illustrated herein is given merely by way of example. However, other methods may also be used to enforce the logical partitioning of the NVRAM. Furthermore, the principals of the present invention may be applied to other devices other than NVRAMs for which logical partitioning is desirable but which do not lend themselves to physical partitioning.

[0037] It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media such a floppy disc, a hard disk drive, a RAM, and CD-ROMs and transmission-type media such as digital and analog communications links.

[0038] The description of the present invention has been presented for purposes of illustration and description, but is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The

embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.

What is claimed is:

1. A method for enforcing logical partitioning of a shared device to which multiple partitions within a data processing system access, the method comprising:

receiving a request from a requesting device to access a portion of the shared device, wherein the request includes a virtual address;

responsive to a determination that the virtual address is included in a range of addresses for which the requesting device is authorized to access, providing access to the requested portion of the shared device to the requesting device.

2. The method as recited in claim 1, further comprising:

responsive to a determination that the virtual address is not included in a range of addresses for which the requesting device may access, denying access to the requested portion of the shared device.

3. The method as recited in claim 1, further comprising:

prior to receiving a request to access a portion of the shared device, assigning each of a plurality of partitions within the data processing system a range of virtual addresses to the shared device wherein each virtual address corresponds to a physical address within the shared device.

4. The method as recited in claim 1, wherein the shared device is a non-volatile random access memory.

5. The method as recited in claim 1, wherein the requesting device is a processor.

6. The method as recited in claim 1, wherein the determination as to whether the virtual address is within a range of addresses for which the requesting device is allowed access comprises determining the partition identification of the requesting device and an address range corresponding to the partition identification.

7. The method as recited in claim 1, further comprising:

converting the virtual address into a physical address; and

performing the requested access.

8. The method as recited in claim 1, wherein the requested access is a read operation.

9. The method as recited in claim 1, wherein the requested access is a write operation.

10. A computer program product in a computer readable media for use in a data processing system for enforcing logical partitioning of a shared device to which multiple partitions within the data processing system access, the computer program product comprising:

first instructions for receiving a request from a requesting device to access a portion of the shared device, wherein the request includes a virtual address;

second instructions, responsive to a determination that the virtual address is included in a range of addresses for which the requesting device is authorized to access, for providing access to the requested portion of the shared device to the requesting device.

11. The computer program product as recited in claim 10, further comprising:

third instructions, responsive to a determination that the virtual address is not included in a range of addresses for which the requesting device may access, for denying access to the requested portion of the shared device.

12. The computer program product as recited in claim 10, further comprising:

third instructions, prior to receiving a request to access a portion of the shared device, for assigning each of a plurality of partitions within the data processing system a range of virtual addresses to the shared device wherein each virtual address corresponds to a physical address within the shared device.

13. The computer program product as recited in claim 10, wherein the shared device is a non-volatile random access memory.

14. The computer program product as recited in claim 10, wherein the requesting device is a processor.

15. The computer program product as recited in claim 10, wherein the determination as to whether the virtual address is within a range of addresses for which the requesting device is allowed access comprises determining the partition identification of the requesting device and an address range corresponding to the partition identification.

16. The computer program product as recited in claim 10, further comprising:

third instructions for converting the virtual address into a physical address; and

fourth instructions for performing the requested access.

17. The computer program product as recited in claim 10, wherein the requested access is a read operation.

18. The computer program product as recited in claim 10, wherein the requested access is a write operation.

19. A system in a computer readable media for use in a data processing system for enforcing logical partitioning of a shared device to which multiple partitions within the data processing system access, the system comprising:

an access receiving unit which receives a request from a requesting device to access a portion of the shared device, wherein the request includes a virtual address;

an address verification unit which, responsive to a determination that the virtual address is included in a range of addresses for which the requesting device is authorized to access, provides access to the requested portion of the shared device to the requesting device.

20. The system as recited in claim 19, further comprising:

a access denial unit which, responsive to a determination that the virtual address is not included in a range of addresses for which the requesting device may access, denies access to the requested portion of the shared device.

21. The system as recited in claim 19, further comprising:

a partitioning assignment unit which, prior to receiving a request to access a portion of the shared device, assigns each of a plurality of partitions within the data processing system a range of virtual addresses to the shared device wherein each virtual address corresponds to a physical address within the shared device.

22. The system as recited in claim 19, wherein the shared device is a non-volatile random access memory.

23. The system as recited in claim 19, wherein the requesting device is a processor.

24. The system as recited in claim 19, wherein the determination as to whether the virtual address is within a range of addresses for which the requesting device is allowed access comprises determining the partition identification of the requesting device and an address range corresponding to the partition identification.

25. The system as recited in claim 19, further comprising:

a conversion unit which converts the virtual address into a physical address; and an access performance unit which performs the requested access.

26. The system as recited in claim 19, wherein the requested access is a read operation.

27. The system as recited in claim 19, wherein the requested access is a write operation.

* * * * *

US 20020124166A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0124166 A1

Lee et al. (43) Pub. Date: Sep. 5, 2002

(54) MECHANISM TO SAFELY PERFORM SYSTEM FIRMWARE UPDATE IN LOGICALLY PARTITIONED (LPAR) MACHINES

(75) Inventors: Van Hoa Lee, Cedar Park, TX (US); Saylleela Nulu, Austin, TX (US)

Correspondence Address:
Duke W. Yee
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380 (US)

(73) Assignee: International Business Machines Corporation, Armonk, NY (US)

(21) Appl. No.: 09/798,164

(22) Filed: Mar. 1, 2001

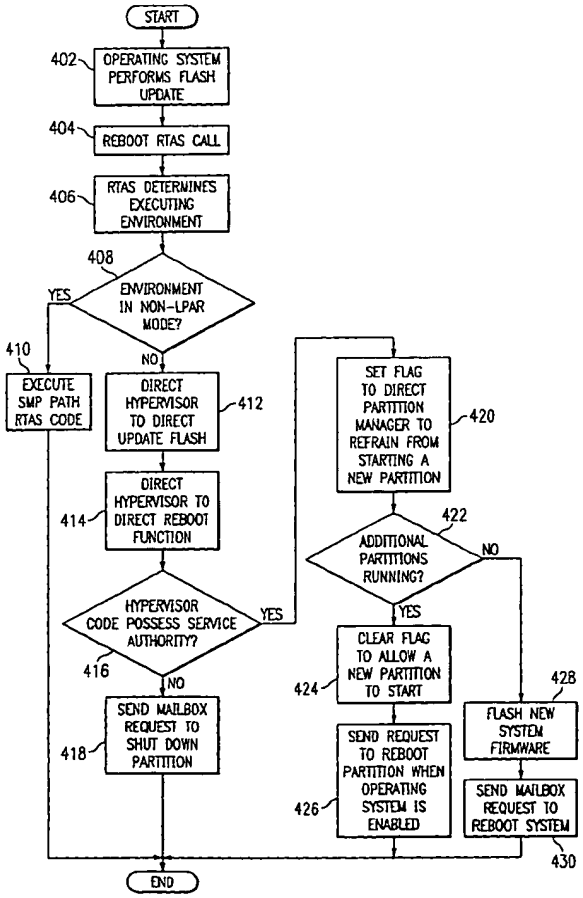
Publication Classification

(51) Int. Cl.[7] G06F 1/24

(52) U.S. Cl. 713/100

(57) ABSTRACT

A method for managing system firmware in a data processing system having a plurality of logical partitions is provided. Responsive to a request to update the system firmware from a first logical partition within the plurality of logical partitions in the data processing system, a determination is made whether the first logical partition within the plurality of logical partitions is present in the data processing system. Responsive to the determination that the first logical partition within the plurality of logical partitions is present in the data processing system, the system firmware is updated from the first logical partition in the data processing system. Then starting of additional partitions within the plurality of logical partitions in the data processing system is inhibited until the firmware update from the first logical partition is complete.



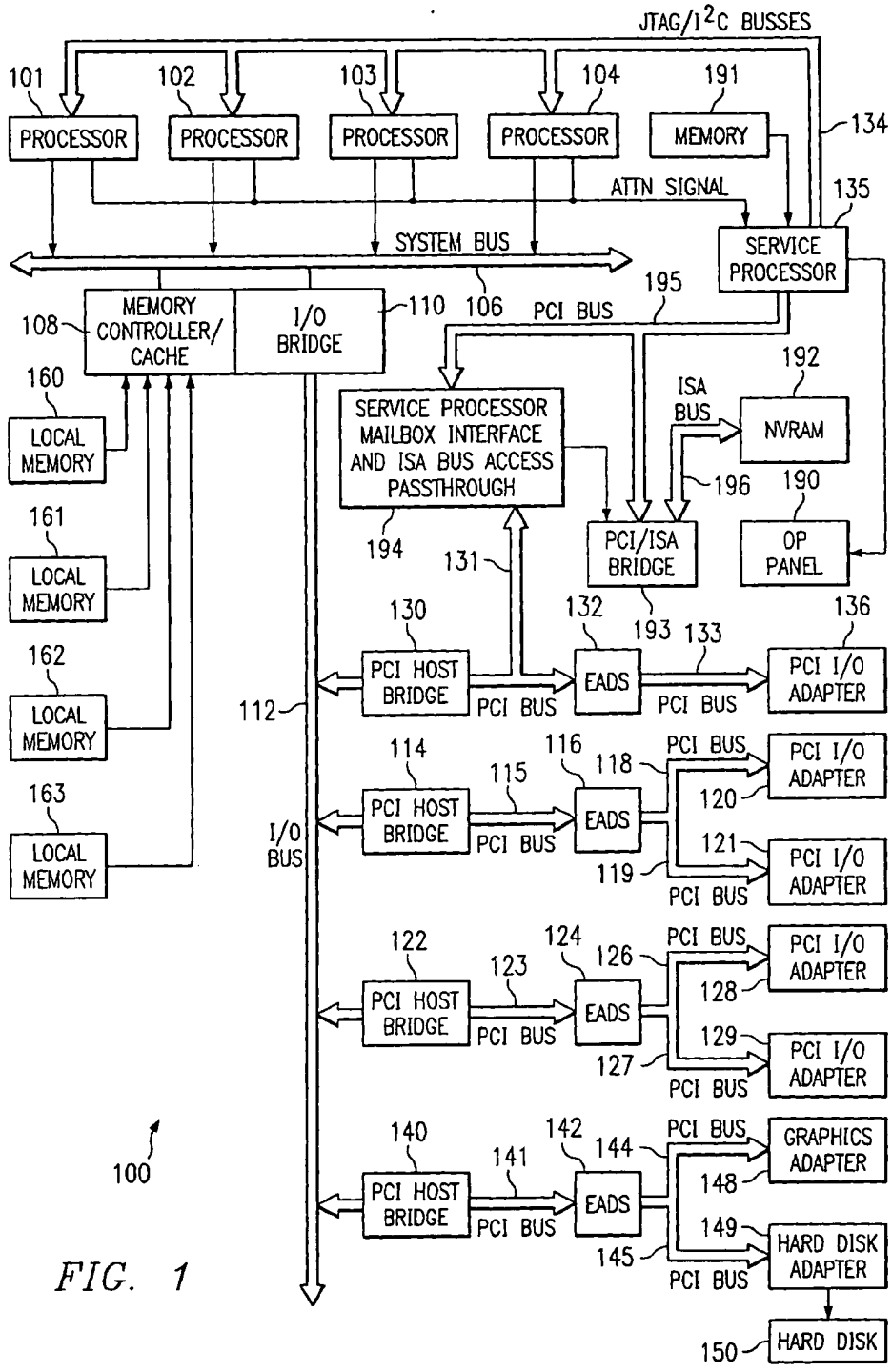

JTAG/I2C BUSSES
101
102
103
104
191
PROCESSOR
PROCESSOR
PROCESSOR
PROCESSOR
MEMORY
134
ATTN SIGNAL
135
SYSTEM BUS
SERVICE PROCESSOR
195
108
MEMORY CONTROLLER/ CACHE
I/O BRIDGE
110
106
PCI BUS
192
160
ISA BUS
SERVICE PROCESSOR MAILBOX INTERFACE AND ISA BUS ACCESS PASSTHROUGH
NVRAM
LOCAL MEMORY
196
190
161
194
131
PCI/ISA BRIDGE
OP PANEL
LOCAL MEMORY
130
132
193
133
136
162
PCI HOST BRIDGE
EADS
PCI I/O ADAPTER
LOCAL MEMORY
112
PCI BUS
PCI BUS
114
116
PCI BUS
163
115
118
PCI HOST BRIDGE
EADS
PCI I/O ADAPTER
LOCAL MEMORY
I/O BUS
120
PCI BUS
121
119
PCI I/O ADAPTER
PCI BUS
122
124
PCI BUS
123
126
PCI HOST BRIDGE
EADS
PCI I/O ADAPTER
128
PCI BUS
129
127
PCI I/O ADAPTER
PCI BUS
140
142
PCI BUS
141
144
GRAPHICS ADAPTER
100
PCI HOST BRIDGE
EADS
148
PCI BUS
149
HARD DISK ADAPTER
145
PCI BUS
150
HARD DISK

FIG. 1

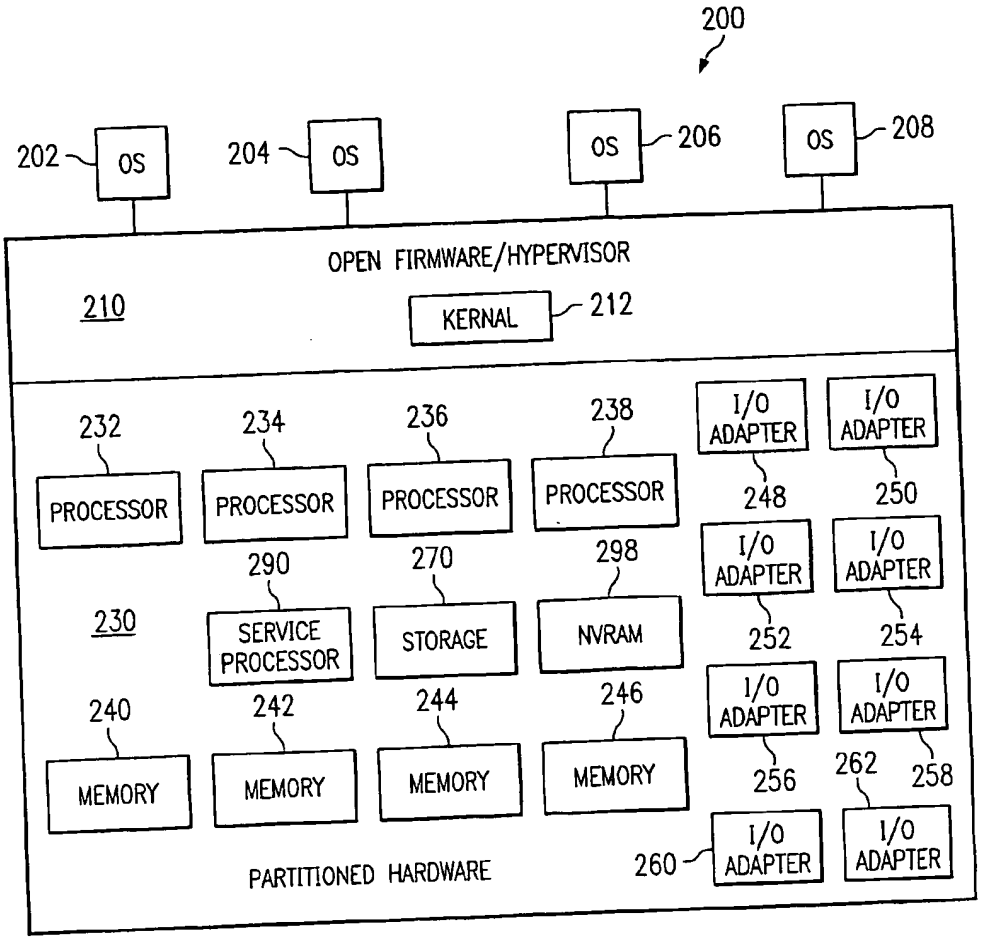
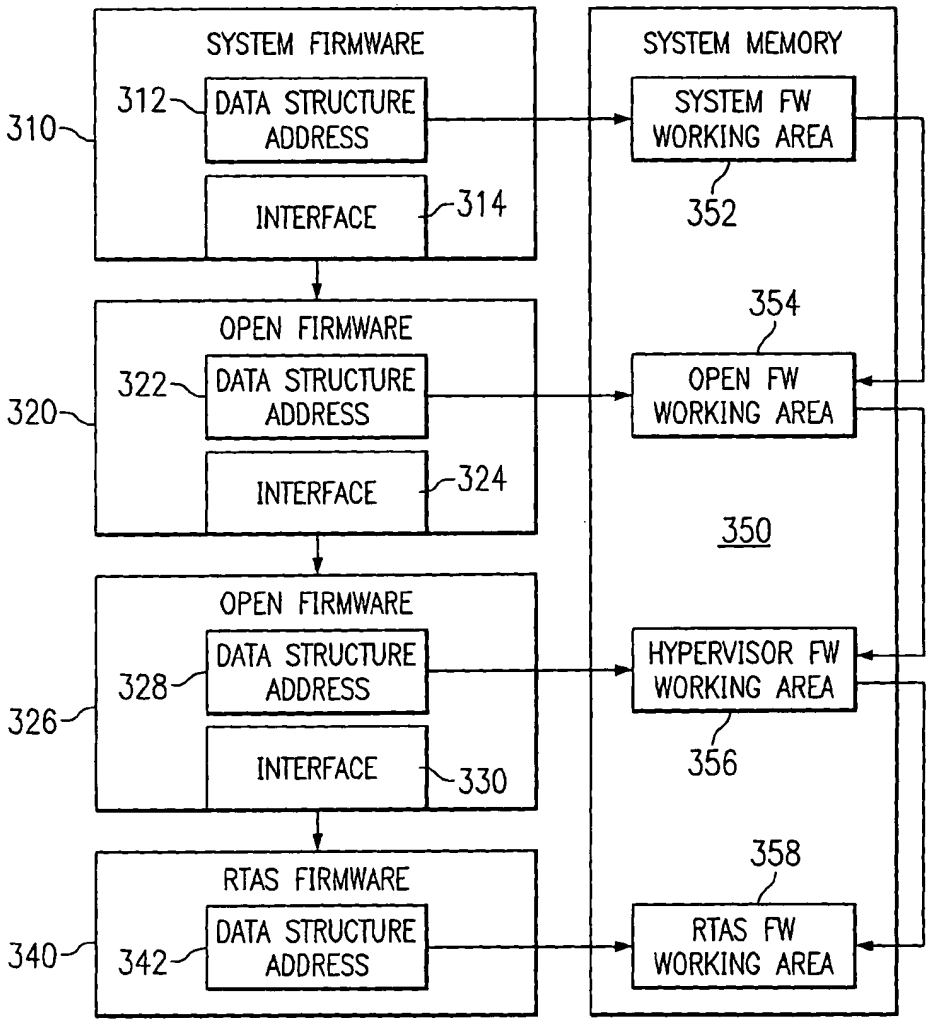

200
202
OS
204
OS
OS
206
OS
208
OPEN FIRMWARE/HYPERVISOR
210
KERNAL
212
232
234
236
238
I/O ADAPTER
I/O ADAPTER
PROCESSOR
PROCESSOR
PROCESSOR
PROCESSOR
248
250
I/O ADAPTER
I/O ADAPTER
290
270
298
230
SERVICE PROCESSOR
STORAGE
NVRAM
252
254
I/O ADAPTER
I/O ADAPTER
240
242
244
246
MEMORY
MEMORY
MEMORY
MEMORY
256
262
258
I/O ADAPTER
I/O ADAPTER
260
PARTITIONED HARDWARE

FIG. 2

FIG. 3


SYSTEM FIRMWARE
310
312
DATA STRUCTURE ADDRESS
INTERFACE
314
OPEN FIRMWARE
320
322
DATA STRUCTURE ADDRESS
INTERFACE
324
OPEN FIRMWARE
326
328
DATA STRUCTURE ADDRESS
INTERFACE
330
RTAS FIRMWARE
340
342
DATA STRUCTURE ADDRESS
SYSTEM MEMORY
SYSTEM FW WORKING AREA
352
354
OPEN FW WORKING AREA
350
HYPERVISOR FW WORKING AREA
356
358
RTAS FW WORKING AREA

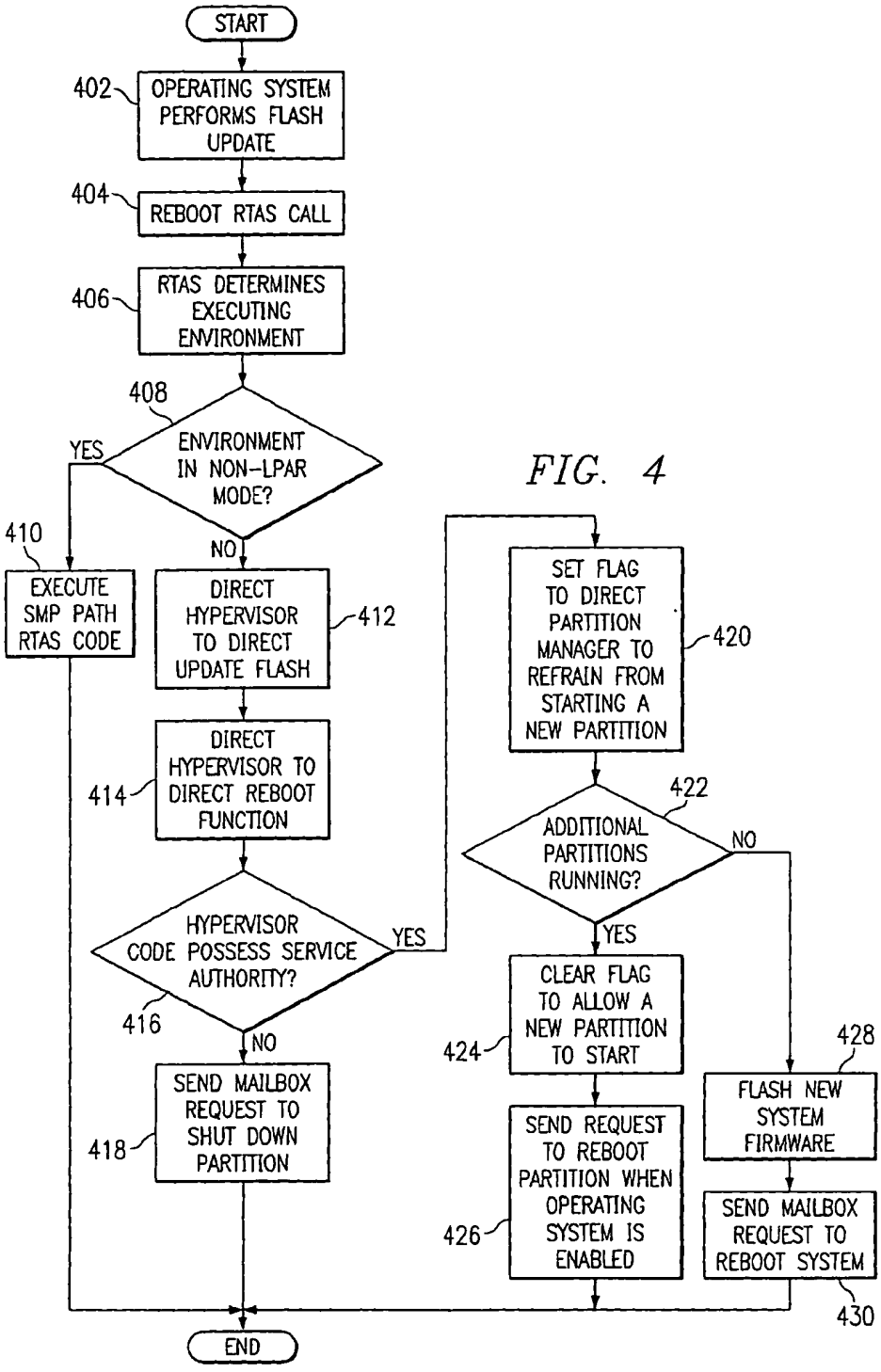

START
402
OPERATING SYSTEM PERFORMS FLASH UPDATE
404
REBOOT RTAS CALL
406
RTAS DETERMINES EXECUTING ENVIRONMENT
408
ENVIRONMENT IN NON-LPAR MODE?
YES
NO
410
EXECUTE SMP PATH RTAS CODE
DIRECT HYPERVISOR TO DIRECT UPDATE FLASH
412
DIRECT HYPERVISOR TO DIRECT REBOOT FUNCTION
414
HYPERVISOR CODE POSSESS SERVICE AUTHORITY?
YES
416
NO
SEND MAILBOX REQUEST TO SHUT DOWN PARTITION
418
SET FLAG TO DIRECT PARTITION MANAGER TO REFRAIN FROM STARTING A NEW PARTITION
420
ADDITIONAL PARTITIONS RUNNING?
422
NO
YES
CLEAR FLAG TO ALLOW A NEW PARTITION TO START
424
SEND REQUEST TO REBOOT PARTITION WHEN OPERATING SYSTEM IS ENABLED
426
428
FLASH NEW SYSTEM FIRMWARE
SEND MAILBOX REQUEST TO REBOOT SYSTEM
430
END

FIG. 4

MECHANISM TO SAFELY PERFORM SYSTEM FIRMWARE UPDATE IN LOGICALLY PARTITIONED (LPAR) MACHINES

BACKGROUND OF THE INVENTION

[0001] 1. Technical Field

[0002] The present invention relates generally to an improved data processing system and in particular to a method and apparatus in a symmetrical multiprocessing system. Still more particularly, the present invention provides a method and apparatus for updating a processor system firmware in a symmetrical multiprocessing system.

[0003] 2. Description of Related Art

[0004] A logical partitioning option (LPAR) within a data processing system (platform) allows multiple copies of a single operating system (OS) or multiple heterogeneous operating systems to be simultaneously run on a single data processing system platform. A partition, within which an operating system image runs, is assigned a non-overlapping subset of the platform's resources. These platform allocable resources include one or more architecturally distinct processors with their interrupt management area, regions of system memory, and I/O adapter bus slots. The partition's resources are represented by its own open firmware device tree to the OS image.

[0005] Each distinct OS or image of an OS running within the platform are protected from each other such that software errors on one logical partition can not affect the correct operation of any of the other partitions. This is provided by allocating a disjoint set of platform resources to be directly managed by each OS image and by providing mechanisms for ensuring that the various images can not control any resources that have not been allocated to it. Furthermore, software errors in the control of an operating system's allocated resources are prevented from affecting the resources of any other image. Thus, each image of the OS (or each different OS) directly controls a distinct set of allocable resources within the platform.

[0006] One method that has been developed to create and maintain separation between the partitions within the data processing system is the use of a firmware component referred to as a hypervisor in the RS/6000 data processing system. The RS/6000 is a product and trademark of International Business Machines Corporation of Armonk, N.Y. This firmware component performs many functions and services for the various operating system images running within the logically partitioned data processing system.

[0007] In the earliest initial power load (IPL) stage, system firmware must perform hardware discovery of the input/output (I/O) subsystem, then initialize and assign system address ranges according to the system memory map for the presenting hardware. Data structures are established and updated when the discovery and initialization are complete.

[0008] Later, in the open firmware stage of the IPL, the open firmware must create its open firmware device tree based on the current hardware in the system. Thus, the open firmware must repeat the discovery of hardware components and store a data structure to describe the hardware.

[0009] Finally, in the last stage of the IPL, the runtime abstraction service (RTAS) firmware needs to know the hardware information so that it can provide services to the operating system during runtime. RTAS also analyzes and isolates hardware problems when the system encounters some error exceptions during runtime. RTAS must also perform the discovery process and create a data structure to describe the hardware.

[0010] With the need for more and more processing power, symmetrical multiprocessing (SMP) systems are being used more often. SMP is a computer architecture in which multiple processors share the same memory, containing one copy of the operating system, one copy of any applications that are in use, and one copy of the data. SMP reduces transaction time because the operating system divides the workload into tasks and assigns those tasks to whatever processors are free.

[0011] SMP systems often times experience failures. Sometimes these failures are so-called hard or solid errors, from which no recovery is possible. A hard error in a SMP system, in general, causes a system failure. Thereafter, the device that has caused the hard error is replaced. On the other hand, a number of failures are recoverable or so-called soft errors, which occur intermittently and randomly. In contrast to a hard error, a soft error, with proper recovery and retry design, can be recovered and prevent a SMP system from failing. Often these soft errors are localized to a particular processor within a SMP system.

[0012] When a Regatta machine is configured to run in the LPAR mode, an update of the System Firmware has to be performed without causing data loss to any running partition of the machine. One method to avoid data loss to any running partition is to reboot the system in a non-LPAR or SMP mode and then perform the update process. However, this may not be desirable since running partitions have to be shut down for the reboot to occur. Therefore, it would be advantageous to have a method and apparatus for updating system firmware when multiple partitions are run to reboot in LPAR mode wherein the partition with service authority updates the firmware.

SUMMARY OF THE INVENTION

[0013] The present invention provides a method for managing system firmware in a data processing system having a plurality of logical partitions. Responsive to a request to update the system firmware from a first logical partition within the plurality of logical partitions in the data processing system, a determination is made whether the first logical partition within the plurality of logical partitions is present in the data processing system. Responsive to the determination that the first logical partition within the plurality of logical partitions is present in the data processing system, the system firmware is updated from the first logical partition in the data processing system. Then starting of additional partitions within the plurality of logical partitions in the data processing system is inhibited until the firmware update from the first logical partition is complete.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further

objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

[0015] **FIG. 1** depicts a block diagram of a data processing system in which the present invention may be implemented;

[0016] **FIG. 2** depicts a block diagram of an exemplary logically partitioned platform in which the present invention may be implemented;

[0017] **FIG. 3** is a block diagram of a firmware arrangement in accordance with a preferred embodiment of the present invention; and

[0018] **FIG. 4** illustrates a flowchart of the operation of a updating firmware in a logically partitioned machine in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0019] With reference now to the figures, and in particular with reference to **FIG. 1**, a block diagram of a data processing system in which the present invention may be implemented is depicted. Data processing system **100** may be a symmetric multiprocessor (SMP) system including a plurality of processors **101**, **102**, **103**, and **104** connected to system bus **106**. For example, data processing system **100** may be an IBM RS/6000, a product of International Business Machines Corporation in Armonk, N.Y., implemented as a server within a network. Alternatively, a single processor system may be employed. Also connected to system bus **106** is memory controller/cache **108**, which provides an interface to a plurality of local memories **160-163**. I/O bus bridge **110** is connected to system bus **106** and provides an interface to I/O bus **112**. Memory controller/cache **108** and I/O bus bridge **110** may be integrated as depicted.

[0020] Data processing system **100** is a logically partitioned data processing system. Thus, data processing system **100** may have multiple heterogeneous operating systems (or multiple instances of a single operating system) running simultaneously. Each of these multiple operating systems may have any number of software programs executing within in it. Data processing system **100** is logically partitioned such that different I/O adapters **120-121**, **128-129**, **136**, and **148-149** may be assigned to different logical partitions.

[0021] Thus, for example, suppose data processing system **100** is divided into three logical partitions, P1, P2, and P3. Each of I/O adapters **120-121**, **128-129**, **136**, and **148-149**, each of processors **101-104**, and each of local memories **160-164** is assigned to one of the three partitions. For example, processor **101**, memory **160**, and I/O adapters **120**, **128**, and **129** may be assigned to logical partition P1; processors **102-103**, memory **161**, and I/O adapters **121** and **136** may be assigned to partition P2; and processor **104**, memories **162-163**, and I/O adapters **148-149** may be assigned to logical partition P3.

[0022] Each operating system executing within data processing system **100** is assigned to a different logical partition. Thus, each operating system executing within data processing system **100** may access only those I/O units that are within its logical partition. Thus, for example, one instance of the Advanced Interactive Executive (AIX) operating system may be executing within partition P1, a second instance (image) of the AIX operating system may be executing within partition P2, and a Windows 2000 operating system may be operating within logical partition P1. Windows 2000 is a product and trademark of Microsoft Corporation of Redmond, Wash.

[0023] Peripheral component interconnect (PCI) Host bridge **114** connected to I/O bus **112** provides an interface to PCI local bus **115**. A number of Input/Output (I/O) adapters **120-121** may be connected to PCI bus **115**. Typical PCI bus implementations will support between four and eight I/O adapters (i.e. expansion slots for add-in connectors). Each I/O Adapter **120-121** provides an interface between data processing system **100** and input/output devices such as, for example, other network computers, which are clients to data processing system **100**.

[0024] An additional PCI host bridge **122** provide an interface for an additional PCI bus **123**. PCI bus **123** is connected to a plurality of PCI I/O adapters **128-129** by a PCI bus **126-127**. Thus, additional I/O devices, such as, for example, modems or network adapters may be supported through each of PCI I/O adapters **128-129**. In this manner, data processing system **100** allows connections to multiple network computers.

[0025] A memory mapped graphics adapter **148** may be connected to I/O bus **112** through PCI Host Bridge **140** and EADS **142** (PCI-PCI bridge) via PCI buses **141** and **144** as depicted. Also, a hard disk **150** may also be connected to I/O bus **112** through PCI Host Bridge **140** and EADS **142** via PCI buses **141** and **145** as depicted. EADS is the IBM internal name of a chip which provides **8** PCI-PCI bridges that support hot plugging of PCI adapters on the secondary buses.

[0026] A PCI host bridge **130** provides an interface for a PCI bus **131** to connect to I/O bus **112**. PCI bus **131** connects PCI host bridge **130** to the service processor mailbox interface and ISA bus access pass-through logic **194** and EADS **132**. The ISA bus access pass-through logic **194** forwards PCI accesses destined to the PCI/ISA bridge **193**. The NV-RAM storage is connected to the ISA bus **196**. The Service processor **135** is coupled to the service processor mailbox interface **194** through its local PCI bus **195**. Service processor **135** is also connected to processors **101-104** via a plurality of JTAG/I^2C buses **134**. JTAG/I^2C buses **134** are a combination of JTAG/scan busses (see IEEE 1149.1) and Phillips I^2C busses. However, alternatively, JTAG/I^2C buses **134** may be replaced by only Phillips I^2C busses or only JTAG/scan busses. All SP-ATTN signals of the host processors **101**, **102**, **103**, and **104** are connected together to an interrupt input signal of the service processor. The service processor **135** has its own local memory **191**, and has access to the hardware op-panel **190**. JTAG stands for Joint Test Action Group. This group created the foundation for IEEE 1149.1 standard describing the Test Access Port and Boundary Scan Architecture. I^2C stands for Inter-IC which is a bus interface specification invented by Philips.

[0027] When data processing system **100** is initially powered up, service processor **135** uses the JTAG/scan buses **134** to interrogate the system (Host) processors **101-104**, memory controller **108**, and I/O bridge **110**. At completion

of this step, service processor **135** has an inventory and topology understanding of data processing system **100**. Service processor **135** also executes Built-In-Self-Tests (BISTs), Basic Assurance Tests (BATs), and memory tests on all elements found by interrogating the system processors **101-104**, memory controller **108**, and I/O bridge **110**. Any error information for failures detected during the BISTs, BATs, and memory tests are gathered and reported by service processor **135**.

[0028] If a meaningful/valid configuration of system resources is still possible after taking out the elements found to be faulty during the BISTs, BATs, and memory tests, then data processing system **100** is allowed to proceed to load executable code into local (Host) memories **160-163**. Service processor **135** then releases the Host processors **101-104** for execution of the code loaded into Host memory **160-163**. While the Host processors **101-104** are executing code from respective operating systems within the data processing system **100**, service processor **135** enters a mode of monitoring and reporting errors. The type of items monitored by service processor include, for example, the cooling fan speed and operation, thermal sensors, power supply regulators, and recoverable and non-recoverable errors reported by processors **101-104**, memories **160-163**, and bus-bridge controller **110**.

[0029] Service processor **135** is responsible for saving and reporting error information related to all the monitored items in data processing system **100**. Service processor **135** also takes action based on the type of errors and defined thresholds. For example, service processor **135** may take note of excessive recoverable errors on a processor's cache memory and decide that this is predictive of a hard failure. Based on this determination, service processor **135** may mark that resource for deconfiguration during the current running session and future Initial Power Loads (IPLs). IPLs are also sometimes referred to as a "boot" or "bootstrap".

[0030] Those of ordinary skill in the art will appreciate that the hardware depicted in **FIG. 1** may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

[0031] **FIG. 2** is a block diagram of an exemplary logically partitioned platform is depicted in which the present invention may be implemented. The hardware in logically partitioned platform **200** may be implemented as, for example, server **100** in **FIG. 1**. Logically partitioned platform **200** includes partitioned hardware **230**, Open Firmware/Hypervisor **210**, and operating systems **202-208**. Operating systems **202-208** may be multiple copies of a single operating system or multiple heterogeneous operating systems simultaneously run on platform **200**.

[0032] Partitioned hardware **230** includes a plurality of processors **232-238**, a plurality of system memory units **240-246**, a plurality of input/output (I/O) adapters **248-262**, and a storage unit **270**. Each of the processors **242-248**, memory units **240-246**, NV-RAM storage **298**, and I/O adapters **248-262** may be assigned to one of multiple partitions within logically partitioned platform **200**, each of which corresponds to one of operating systems **202-208**.

[0033] Open Firmware/Hypervisor **210** performs a number of functions and services for operating system images **202-208** to create and enforce the partitioning of logically partitioned platform **200**. Firmware is "software" stored in a memory chip that holds its content without electrical power, such as, for example, read-only memory (ROM), programmable ROM (PROM), erasable programmable ROM (EPROM), electrically erasable programmable ROM (EEPROM), and non-volatile random access memory (non-volatile RAM).

[0034] Open Firmware/Hypervisor **210** is a firmware implemented virtual machine identical to the underlying hardware. Thus, Open Firmware/Hypervisor **210** allows the simultaneous execution of independent OS images **202-208** by virtualizing all the hardware resources of logically partitioned platform **200**. Open Firmware/Hypervisor **210** may attach I/O devices through I/O adapters **248-262** to single virtual machines in an exclusive mode for use by one of OS images **202-208**.

[0035] At startup, in the earliest initial power load (IPL) stage, the system firmware must perform the hardware discovery of the I/O subsystem, then initialize and assign system address ranges according to the system memory map for the presenting hardware. Data structures are established and updated when the discovery and initialization are complete.

[0036] The present invention provides a mechanism to support flash_update and reboot RTAS call from an operating system to perform system firmware update in a LPAR capable machine. In LPAR mode, the mechanism is carried out in a hypervisor code. The hypervisor may enforce protection to make sure that the partition with service authority only performs this specific function. Before the mechanism is to be executed, a partition manager may be informed not to start any new partitions so that the service authorized partition may complete the system firmware update and safely reboot the machine.

[0037] **FIG. 3** is a block diagram of a firmware arrangement in accordance with a preferred embodiment of the present invention. System firmware (FW) **310** performs hardware discovery in the earliest IPL stage and creates data structures in system firmware working area **352** of system memory **350**.

[0038] In the open firmware stage of the IPL, system firmware **310** provides data structure address **312** to open firmware **320** through interface (IF) **314**. Open firmware **320** then creates a copy of the data structures from system firmware **310** into open firmware working area **354** of system memory **350**.

[0039] In the hypervisor firmware stage of the IPL, open firmware **320** provides data structure address **328** to hypervisor firmware **326** through interface **330**. Hypervisor firmware **326** then creates a copy of the data structures from open firmware **320** into hypervisor firmware working area **356** of system memory **350**.

[0040] When a RTAS firmware component is instantiated, hypervisor firmware **326** provides data structure address **328** to RTAS firmware **340** through interface **330**. RTAS firmware **340** then creates a copy of the data structures from hypervisor firmware **326** into RTAS firmware working area **358** of system memory **350**.

[0041] **FIG. 4** illustrates a flowchart of the operation of a updating firmware in a logically partitioned machine in

accordance with a preferred embodiment of the present invention. In this example, the operation begins by an operating system performing a flash update (step **402**). Then a reboot is made by a RTAS call (step **404**). The RTAS then determines the executing environment (step **406**). Then a determination is made as to whether or not the executing environment is in a non_LPAR mode (step **408**). If the executing environment is in a non-LPAR mode (step **408**:YES), then the SMP is executed using a path RTAS code (step **410**) and thereafter the operation terminates.

[0042] If the executing environment is not in a non_lpar mode (step **408**:NO), the hypervisor is directed to direct the flash update (step **412**). The hypervisor is then directed to direct the reboot function (step **414**). Then a determination is made as to whether or not a hypervisor code possesses service authority (step **416**).

[0043] If the hypervisor code does not possess service authority (step **416**:NO), a mailbox is sent a request to shut down the partition (step **418**) and thereafter the operation terminates. If the hypervisor code does possess service authority (step **416**:YES), a flag is set to direct a partition manager to refrain from starting a new partition (step **420**). Then a determination is made as to whether or not additional partitions are running (step **422**). If additional partitions are not running (step **422**:NO), new system hardware is flashed (step **428**). Then a mailbox request is sent to reboot the system (step **430**) and thereafter the operation terminates. If additional partitions are not running (step **422**:NO), the flag is cleared to allow a new partition to start (step **424**). Then a request is sent to reboot the partition when the operating system is enabled (step **426**) and thereafter the operation terminates.

[0044] Thus, the present invention solves the disadvantages of the prior art by providing a method and apparatus for updating system firmware when multiple partitions are run to reboot in LPAR mode wherein the partition with service authority updates the firmware. An operating system may reboot in LPAR mode when a partition with service authority updates the firmware and no additional partitions may be started until the firmware update occurs.

[0045] It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media such as a floppy disc, a hard disk drive, a RAM, and CD-ROMs and transmission-type media such as digital and analog communications links.

[0046] The description of the present invention has been presented for purposes of illustration and description, but is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.

What is claimed is:

1. A method for managing system firmware in a data processing system having a plurality of logical partitions, the method comprising the steps of:

responsive to a request to update the system firmware from a first logical partition within the plurality of logical partitions in the data processing system, determining whether the first logical partition within the plurality of logical partitions is present in the data processing system;

responsive to determining a first logical partition within the plurality of logical partitions is present in the data processing system, updating the system firmware from the first logical partition in the data processing system; and

inhibiting starting of additional partitions within the plurality of logical partitions in the data processing system until the system firmware update from the first logical partition is complete.

2. The method as recited in claim 1, further comprising:

responsive to completion of the system firmware update from the first logical partition, receiving a request to reboot the data processing system from the first logical partition; and

activating a reboot signal for a processor assigned to the first logical partition.

3. The method as recited in claim 2, wherein rebooting the logical partition is performed by a hypervisor.

4. The method as recited in claim 1, wherein the request to update the system firmware from the first logical partition is received from a remote terminal.

5. The method as recited in claim 1, wherein the request to update the system firmware from the first logical partition is received from a hypervisor.

6. The method as recited in claim 1, wherein updating the system firmware from the first logical partition in the data processing system is performed by a hypervisor.

7. The method as recited in claim 6, wherein the hypervisor ensures that only the first logical partition in the data processing system performs the system firmware update.

8. A data processing system comprising:

a bus system;

a communications unit connected to the bus system;

a memory connected to the bus system, the memory including a set of instructions; and

a processor connected to the bus system, wherein the processing unit executes the set of instructions from the memory to determine whether a first logical partition within the plurality of logical partitions is present in the data processing system in response to receiving a request to update the system firmware from a first logical partition within the plurality of partitions, updates the system firmware from the first logical partition in the data processing system in response to determining a first logical partition within the plurality of logical partitions is present in the data processing system, and inhibits starting of additional partitions within the plurality of logical partitions in the data processing system until the system firmware update from the first partition is complete.

9. A data processing system for managing system firmware in a data processing system having a plurality of logical partitions, comprising:

determining means, responsive to a request to update the system firmware from a first logical partition within the plurality of logical partitions in the data processing system, for determining whether a first logical partition within the plurality of logical partitions is present in the data processing system;

updating means, responsive to determining a first logical partition within the plurality of logical partitions is present in the data processing system, for updating the system firmware from the first logical partition in the data processing system; and

inhibiting means for inhibiting starting of additional partitions within the plurality of logical partitions in the data processing system until the system firmware update from the first logical partition is complete.

10. The data processing system as recited in claim 9, further comprising:

receiving means, responsive to completion of the system firmware update from the first logical partition, for receiving a request to reboot the data processing system from the first logical partition; and

activating means for activating a reboot signal for a processor assigned to the first logical partition.

11. The data processing system as recited in claim 10, wherein rebooting the logical partition is performed by a hypervisor.

12. The data processing system as recited in claim 9, wherein the request to update the system firmware from the first logical partition is received from a remote terminal.

13. The data processing system as recited in claim 9, wherein the request to update the system firmware from the first logical partition is received from a hypervisor.

14. The data processing system as recited in claim 9, wherein updating the system firmware from the first logical partition in the data processing system is performed by a hypervisor.

15. The data processing system as recited in claim 14, wherein the hypervisor ensures that only the first logical partition in the data processing system performs the system firmware update.

16. A computer program product in a computer readable medium for managing system firmware in a data processing system having a plurality of logical partitions, comprising:

first instructions, responsive to a request to update the system firmware from a first logical partition within the plurality of logical partitions in the data processing system, for determining whether a first logical partition within the plurality of logical partitions is present in the data processing system;

second instructions, responsive to determining a first logical partition within the plurality of logical partitions is present in the data processing system, for updating the system firmware from the first logical partition in the data processing system; and

third instructions for inhibiting starting of additional partitions within the plurality of logical partitions in the data processing system until the system firmware update from the first logical partition is complete.

17. The computer program product as recited in claim 16, further comprising:

third instructions, responsive to completion of the system firmware update from the first logical partition, for receiving a request to reboot the data processing system from the first logical partition; and

fourth instructions for activating a reboot signal for a processor assigned to the first logical partition.

18. The computer program product as recited in claim 17, wherein rebooting the logical partition is performed by a hypervisor.

19. The computer program product as recited in claim 16, wherein the request to update the system firmware from the first logical partition is received from a remote terminal.

20. The computer program product as recited in claim 16, wherein the request to update the system firmware from the first logical partition is received from a hypervisor.

21. The computer program product as recited in claim 16, wherein updating the system firmware from the first logical partition in the data processing system is performed by a hypervisor.

22. The computer program product as recited in claim 21, wherein the hypervisor ensures that only the first logical partition in the data processing system performs the system firmware update.

* * * * *